系统配置概要、详细设计文档

Revision:	2
Status:	Draft
Author:	Huang Xin
Function:	Netease, Inc.
Path and Filename:	
File Type:	Microsoft Word 2017

Revision History

Rev	Status*	Date	Author	Reason for change/description
1	D	2017/08/04	Huang Xin	First draft
2	D	2017/08/15	Huang Xin	1. 增加增加使用限定说明在第 11 页 2. 修改 CfgGetKeyValueSync 函数名为 CfgGetKeyValueAsync 在第 12 页

* Statuses include: D: Draft, P: Pending review/frozen, U: Update needed, A: Approved

Approved by: **Date:**

1 概述

本文档详细描述智能音箱产品嵌入式软件中使用到的配置项管理方案。实现了配置项的建立，查询，修改，存储的基本功能。以及在此基础上做出的扩展，主要涵盖了高速缓存，异步磁盘 IO 访问，RPC 调用接口等。致力于提供一个通用，可移植，安全，兼容性好，高效，使用简单的配置项功能。该方案是嵌入式软件异步框架的一个基本组成部分。RPC 调用接口依赖于嵌入式异步软件框架的 DBus 消息。

2 术语

术语	说明

3 概要设计

3.1 背景信息

智能音箱采用 AllWinner 公司提供的 R16 Tina Linux 方案，存储介质使用 Nand Flash 大容量存储芯片，采用交流供电，没有备份电源。为了保障掉电的情况下不损坏磁盘分区，Linux 操作系统中没有开启磁盘 IO Cache，这样导致访问磁盘文件异常缓慢。所有系统不能频繁地读写磁盘文件。而配置项可以在断电后继续使用，这就必然要将某些配置信息写入到磁盘中。如何解决这两者之间的冲突就是本方案的设计目的。

3.2 设计目标

3.2.1 高效性

如何更快地添加、修改、获取配置信息。

3.2.2 易用性

如何更简单方便地进行操作，并且能被不同的进程并发使用，能适应绝大多数应用场景，支持绝大多数数据类型。

3.2.3 安全性（扩展）

如何保护隐私，机密数据的安全性。

3.3 技术方案

数据库 + 哈希表 + 进程通信 + 异步 IO [注1] + 加密 + 压缩 [注2]。

3.3.1 数据库 SQLite3

SQLite3 是一个简单，功能强大的嵌入式关系型数据库，也是目前应用最广泛最稳定的嵌入式客户端关系型数据库。它作为一个进程内的库，实现了自给自足的、无服务器的、零配置的、事务性的 SQL 数据库引擎。它具有以下优点：

- 不需要一个单独的服务器进程或操作的系统（无服务器的）。
- SQLite 不需要配置，这意味着不需要安装或管理。

[1] 红色表示基础功能

[2] 绿色表示可选扩展功能

- 一个完整的 SQLite 数据库是存储在一个单一的跨平台的磁盘文件。
- SQLite 是非常小的，是轻量级的，完全配置时小于 400KiB，省略可选功能配置时小于 250KiB。
- SQLite 是自给自足的，这意味着不需要任何外部的依赖。
- SQLite 事务是完全兼容 ACID 的，允许从多个进程或线程安全访问。
- SQLite 支持 SQL92（SQL2）标准的大多数查询语言的功能。
- SQLite 使用 ANSI-C 编写的，并提供了简单和易于使用的 API。
- SQLite 可在 UNIX（Linux, Mac OS-X, Android, iOS）和 Windows（Win32, WinCE, WinRT）中运行。

3.3.2 哈希表 uthash

uthash 是一个类似 Linux 内核 hash 链表的 Hash 表实现。它对 Hash 链表进行了进一步扩展：支持 C 语言任意结构体数据类型加入到 Hash 表中；支持 C 语言整形、字符串、指针、结构体作为 Hash Key 索引；支持对 Hash 表中的数据进行添加、删除、查询、替换、统计、排序、以及遍历操作等。功能强大且简单实用。

3.3.3 进程通信 DBus

D-Bus 是一个进程间通信及远程过程调用机制，可以让多个不同的计算机程序（即进程）在同一台电脑上同时进行通信。

3.3.4 异步 IO libuv

Libuv 是一个专注于异步 IO 的跨平台高性能事件驱动程序库，具有以下主要功能：

- 由 epoll，kqueue，IOCP，事件端口支持的全功能事件循环。
- 异步 TCP 和 UDP 套接字

- 异步 DNS 解析
- 异步文件和文件系统操作
- 文件系统事件通知
- ANSI 转义码控制 TTY
- IPC 与套接字共享，使用 Unix 域套接字或命名管道（Windows）
- 子进程
- 线程池
- 信号处理
- 高分辨率时钟
- 线程和同步原语

3.3.5 加密算法 OpenSSL

利用基于 OpenSSL 的高速对称加密算法 AES 对 SQLite3 数据库本地文件进行加密，避免隐私和敏感数据泄漏。

3.3.6 压缩算法 LZ4

LZ4 是无损压缩算法，提供每个核心压缩速度为 400 MB / s，可扩展多核 CPU。它具有非常快的解码器，每个内核具有多达几个 GB / s 的速度，通常在多核系统上达到 RAM 速度限制。官方给出的测试结果见表 1.1。

Compressor	Ratio	Compression	Decompression
memcpy	**1**	**7300 MB/s**	**7300 MB/s**
LZ4 fast 8 (v1.7.3)	**1.799**	911 MB/s	3360 MB/s
LZ4 default (v1.7.3)	2.101	625 MB/s	3220 MB/s
LZO 2.09	**2.108**	**620 MB/s**	**845 MB/s**
QuickLZ 1.5.0	**2.238**	**510 MB/s**	**600 MB/s**
Snappy 1.1.3	**2.091**	**450 MB/s**	**1550 MB/s**
LZF v3.6	**2.073**	**365 MB/s**	**820 MB/s**
Zstandard 1.1.1 -1	**2.876**	**330 MB/s**	**930 MB/s**
Zstandard 1.1.1 -3	**3.164**	**200 MB/s**	**810 MB/s**
zlib deflate 1.2.8 -1	**2.73**	**100 MB/s**	**370 MB/s**
LZ4 HC -9 (v1.7.3)	2.72	**34 MB/s**	3240 MB/s
zlib deflate 1.2.8 -6	**3.099**	**33 MB/s**	**390 MB/s**

表 1.1：LZ4 性能测试对比

3.4 软件框架

系统采用分成设计，主要分为接口层、高速缓存、数据引擎，持久化存储四层。一个进程的配置项由私有配置表配置项，公共配置表的配置项以及进程的临时配置项（只存在于高速缓存中的配置项，掉电后丢失）组成。

3.4.1 接口层

主要提供 SDK 调用库和一个全局公用配置管理引擎，该引擎提供了一组 RPC API 来供其它进程对全局公共配置进行操作。

3.4.2 高速缓存

系统运行的核心，应用层所有操作都在高速引擎中进行。高速缓存采用 Key-Value 的键值对应关系设计。Key 在整个系统中必须保持唯一性注3。Value 支持整型、浮点数、字符串三种基础类型，其它数据类型例如字符，无符号数，布尔型等需要转化成以上三种基本类型再加入到系统中。如果不能转换成三种基本类型的数据不被支持注4。每个进程的启动的时候，高速缓存会从数据引擎中加载公共配置项和私有配置项到高速缓存中。

3.4.3 数据引擎

系统的轴心，它向上提供高速缓存使用的原始数据，向下控制数据的持久化存储和数据同步操作。数据引擎基于 SQLite3 数据库，数据库为每一个进程提供了一个私有的配置项表和一个公共的配置项表。数据引擎主要提供数据的存取操作和多进程操作的同步原语。数据引擎还维持一个定时任务用于检测高速缓存中是否有新的数据需要和数据库同步。数据引擎还提供安全性需要的加密、解密以及压缩、解压功能的实现，该实现基于 SQLite3 提供的接口。

3.4.4 持久化存储

配置项的持久化存储由 Linux 文件系统实现。数据引擎的数据库以文件的形式保存在 Linux 文件系统中。Linux 文件系统采用 Nand Flash + JIFFS2 文件系统方案。SQLite3 数据库文件利用异步 IO 方式和文件系统进行交互。

注3：进程中的私有配置项名称不能和公共配置项名相同；各个进程私有配置项名称可以重复。

注4：例如图片，二进制文件，音乐，视频等

4 详细设计

4.1 模型设计

系统采用分层设计，系统整体层次设计见图 2.1。

4.1.1 SDK/API 接口

提供一组 SDK 接口 CfgxxxKeyValue 来进行本地配置的操作；接口层还提供一组 CMD_CFG_xxx 的 RPC API 接口来操作公共配置项。每个进程在启动后都有一份公共配置项的拷贝，所以如果某个进程对公共配置项进行了修改，接口层会调用一个 CMD_CFG_NOTIFY API 通知每一个进程这个修改，每一个进程利用 CMD_CFG_NOTIFY 消息来进行数据同步操作。

CMD_CFG_xxx 集成在嵌入式软件通用框架的 DBus 消息中，进程初始化时调用了 DBusWithLibuvInit 函数后，系统会自动处理 CMD_CFG_xxx 相关消息对配置项进行对应操作，操作结果通过 DBusWithLibuvCfgInit 函数初始化的回调函数通知进程。

为了能统一、正确地使用配置项功能，现对系统做如下约定：

1） 配置项名称根据不同的进程，使用不同的前缀，公共配置项前缀设定为 com(例如：comMacAddr、comCpuId)。

2） 配置项从高速缓存同步到 Flash 上可能有约 1 秒的延时，如果在这个间隙中出现断电、进程异常退出等异常情况，可能导致这个时间段内新增加或修改的结果没有被正确地写回到 Flash 中，导致下次进程重新启动后的配置值还是旧的。

3） 不建议除控制中心外的其它任何进程调用 CMD_CFG_*xxx* 接口在增加、修改配置项。这样可能引起配置项混乱。

4.1.2 Hash Table 高速缓存

进程初始化过程中，调用 CfgGlobalEnvInit 函数对配置功能进行初始化。配置初始化主要有三个步骤：

1) 数据库初始化：

检测数据库是否存在，如果数据库不存在，就新建一个默认的数据库并创建对应的数据表。

2) 加载数据到高速缓存：

依次异步加载公共配置项和私有配置项到 Hash Table 中，SQLite3 数据库中保存了配置项名称，配置项对应的值的字符，以及配置项类型（字符串、整型、浮点型）。加载的过程中需要根据配置项值的类型将字符串进行数值转换。加载内容参见图 2.2

3) 启动高速缓存同步服务：

创建一个新的线程，每秒钟检测一次 Hash Table 高速缓存中是否有需要同步到数据库中的修改项。如果有修改需要同步到数据库，以 SQL 事务的方式向数据库进行同步。这样能减少磁盘 IO 和保证数据的完整性。

4.1.3 配置项组成

进程的配置项分为两类：临时配置项和持久配置项。其中持久配置项又分为进程私有配置项和系统公共配置项。进程私有配置项和临时配置项只有进程自身使用，其它进程不能访问。系统公共配置项对系统中所有进程可用，每个进程都有一份拷贝，并且利用 CMD_CFG_NOFITY 消息进行同步配置项内容。由于消息通知是异步的，导致系统公共配置项在进程中的值可能不是正确的值，所以比较重要的配置项最好是调用 CfgGetKeyValueAsync 进行获取而不是使用 CfgGetKeyValue 在本进程缓存中获取注5。系统配置项组成参见图 2.3

注5：系统公共配置项基本上都是只读的，比如设备信息，生产信息，硬件信息等。将这些公共配置加载到每个进程的缓存中能大大加快访问速度，并且不用等异步返回消息。但是某些公共配置项，比如用户名、密码等可能随时会被修改，这个时候需要应用程序来区分具体是从缓存中获取还是利用 RPC API 从配置服务中获取最新的值。

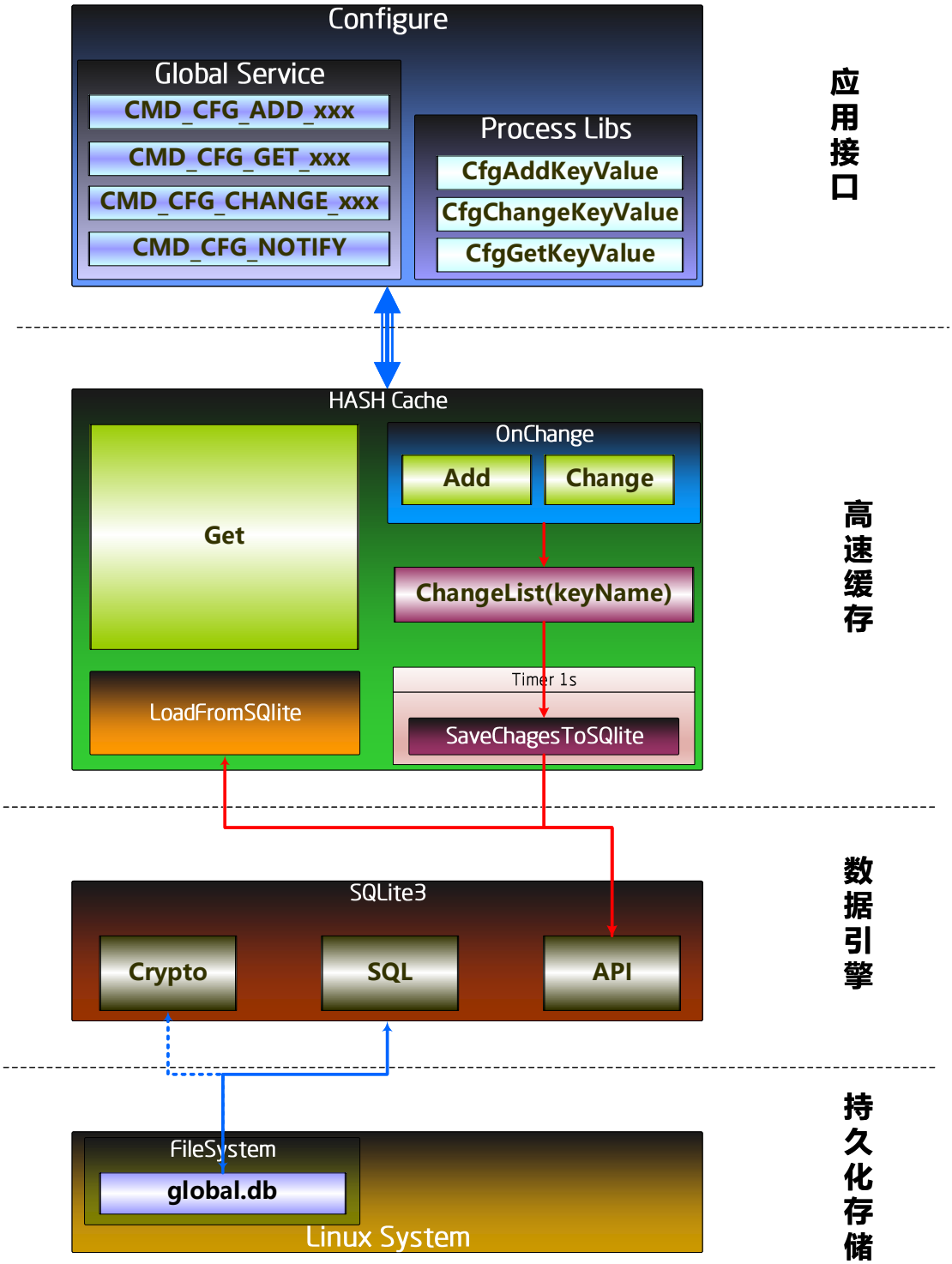

图2.1： 系统层次模型

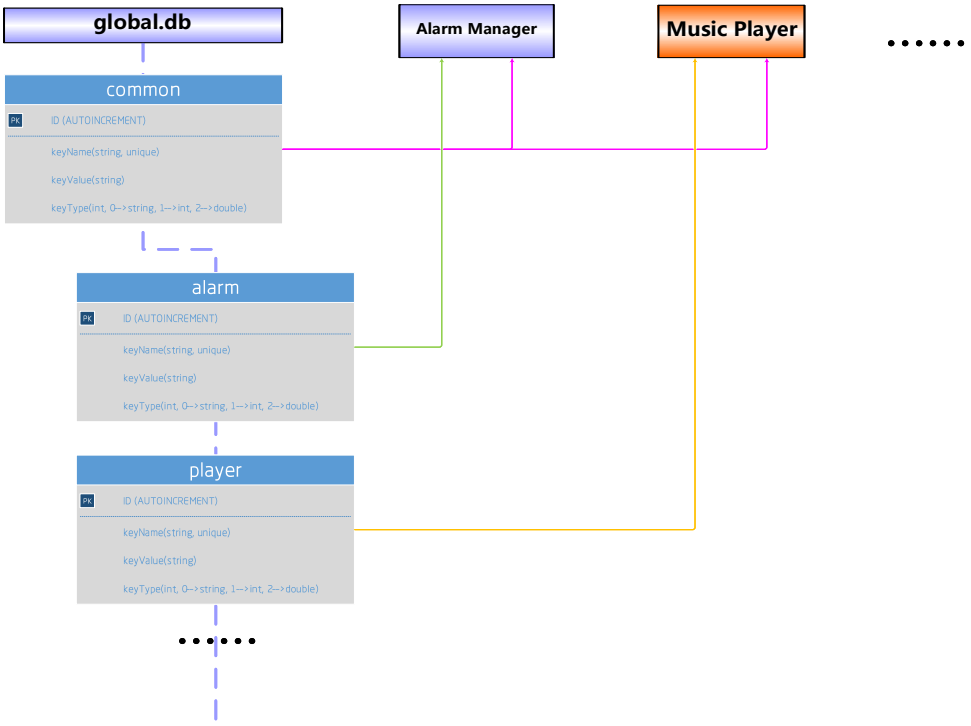

图2.2：　　进程配置内容加载示意图

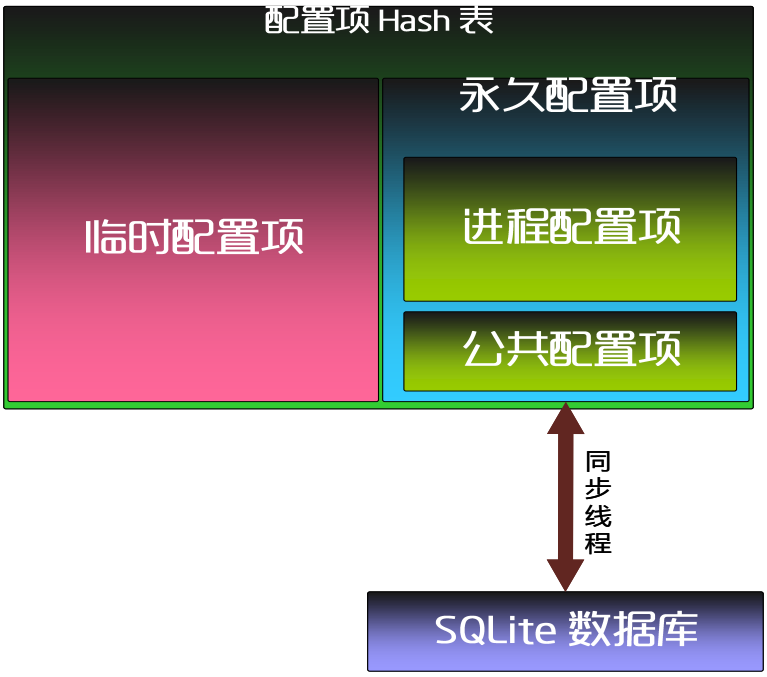

图2.3：　　进程配置项组成示意图

4.2 交互逻辑

系统交互逻辑参见图 2.3

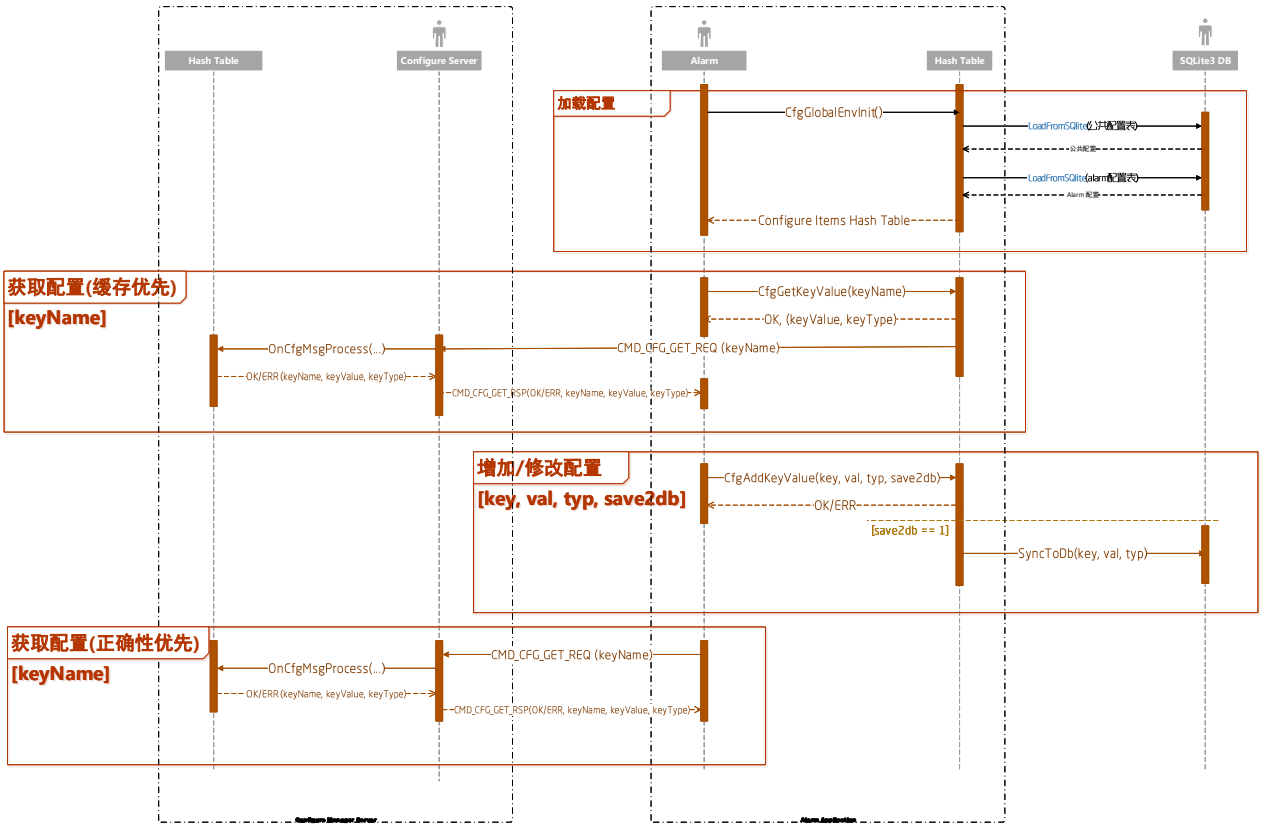

图2.4： 数据交互逻辑

4.3 主要功能流程图

4.3.1 MainInit()

进程初始化函数，触发系统进行相关配置。

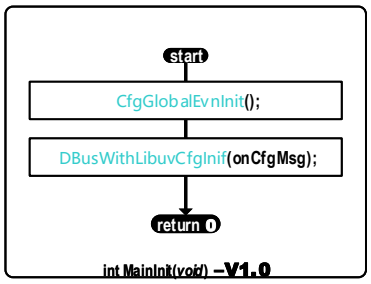

图2.5： 初始化流程图

4.3.2 CfgGlobalEnvInit()

配置内容初始化函数。

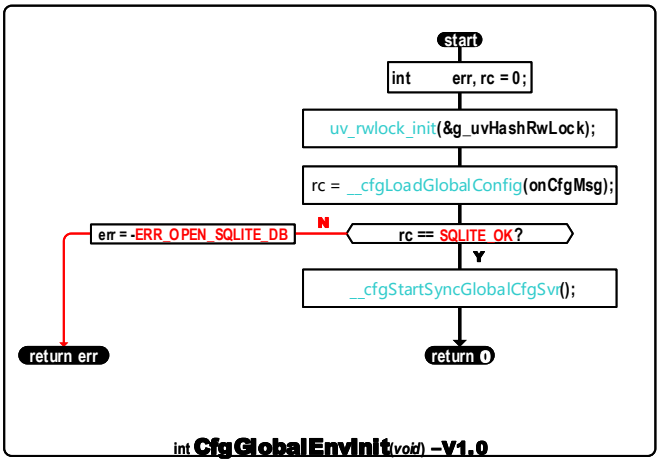

图2.6： 配置初始化流程图

4.3.3 OnCfgMsgProcess()

配置 RPC API 消息处理函数。

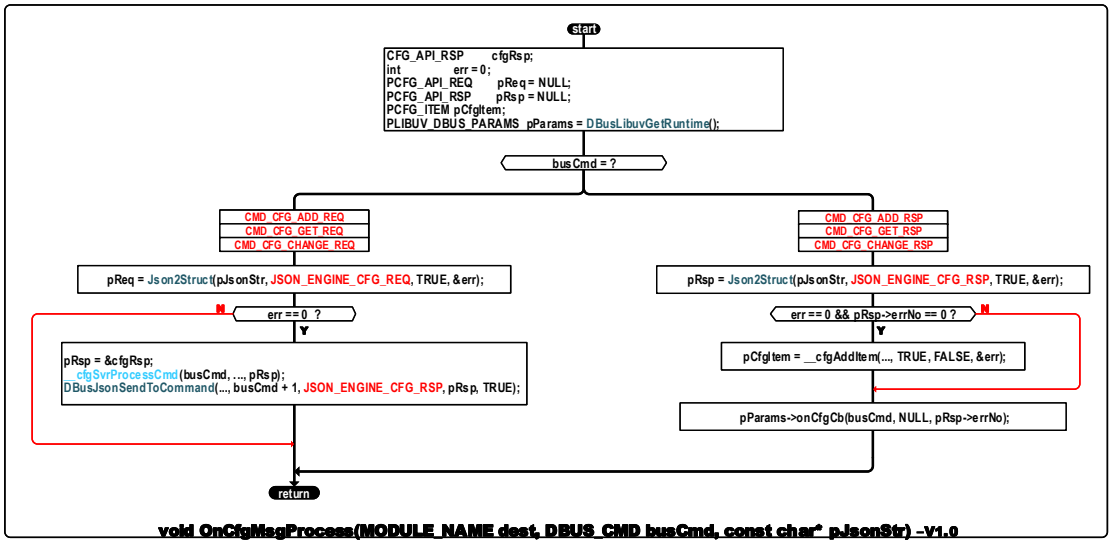

图2.7： RPC API 消息处理流程图

4.3.4 CfgAddKeyValue()

增加一个本地临时/永久配置项。

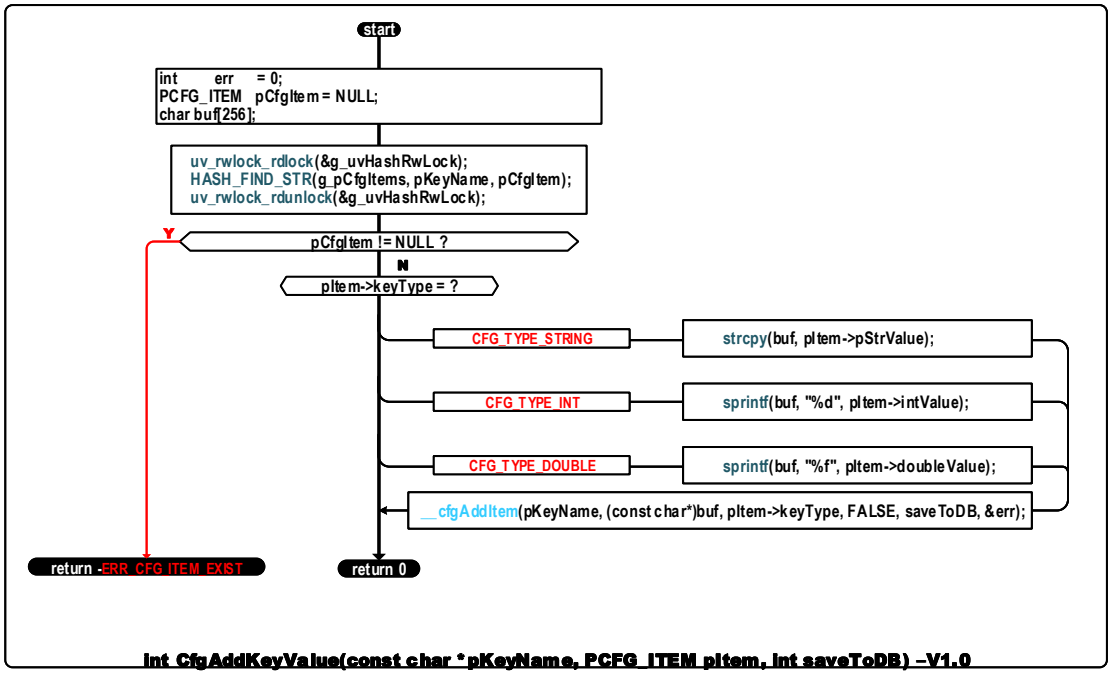

图2.8： 增加配置项流程图

4.3.5 CfgGetKeyValue()

从进程上下文获取一个配置项的值，如果当前进程上下文不存在，自动从全局公共配置项服务中获取。

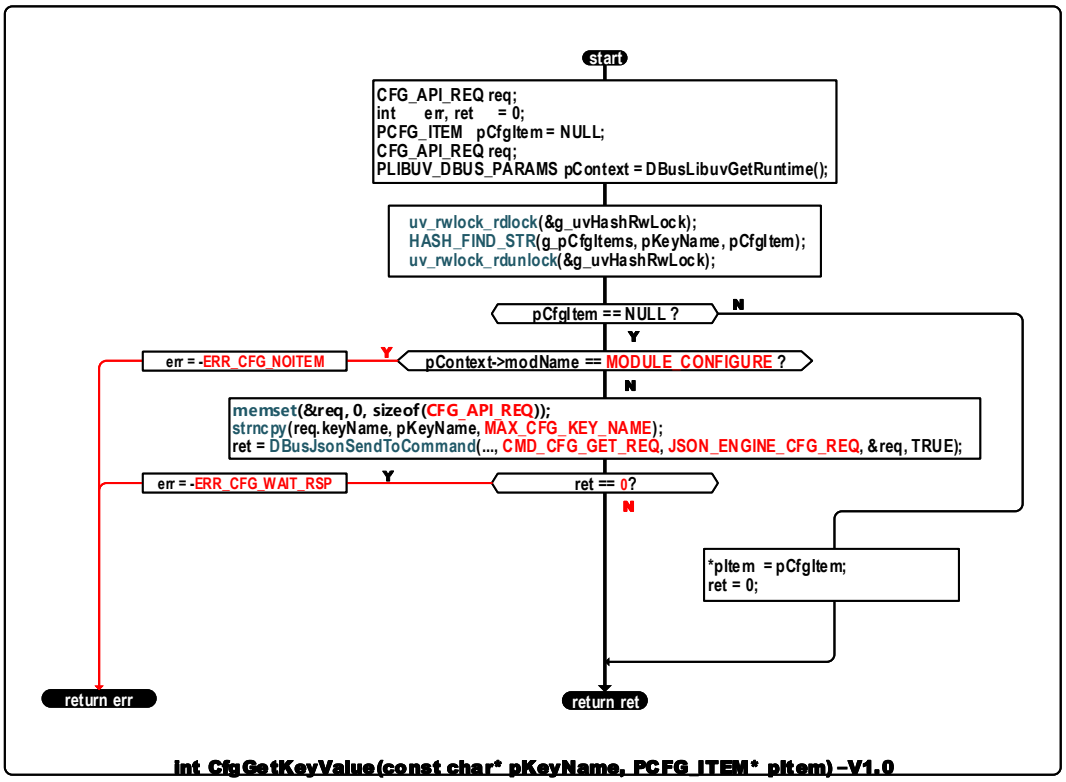

图2.9： 查询配置项值流程图

4.3.6 __cfgLoadGlobalConfig()

加载配置数据库。

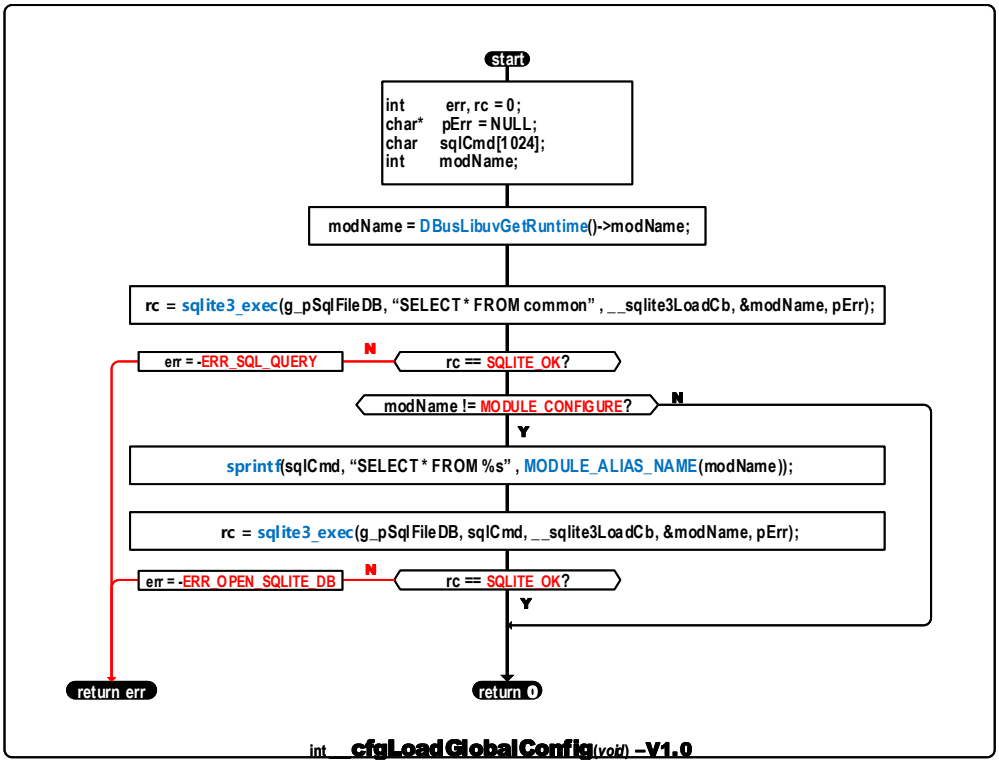

图2.10： 本地数据库初始化流程图

4.3.7 __sqlite3LoadCb()

从数据库加载配置项到高速缓存。

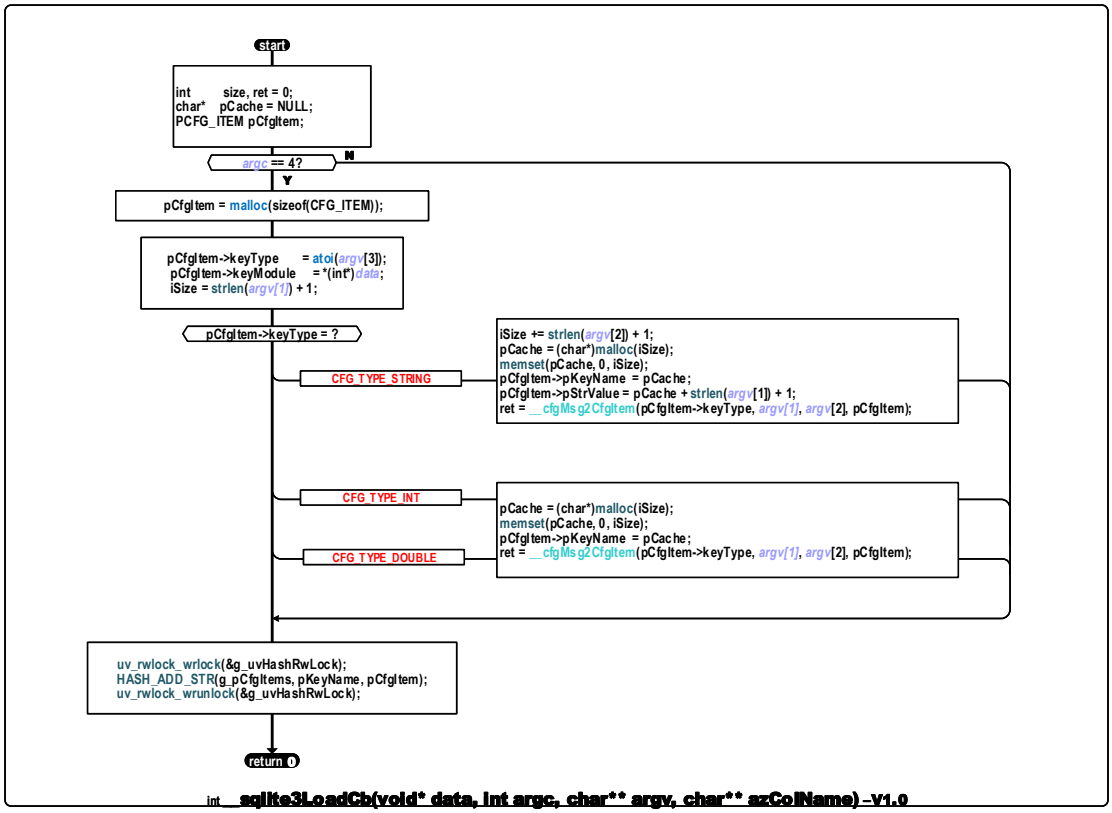

图2.11： 加载配置项到高速缓存流程图

4.3.8 __cfgMsg2CfgItemV2()

RPC API 消息和配置项数据结构转换函数。

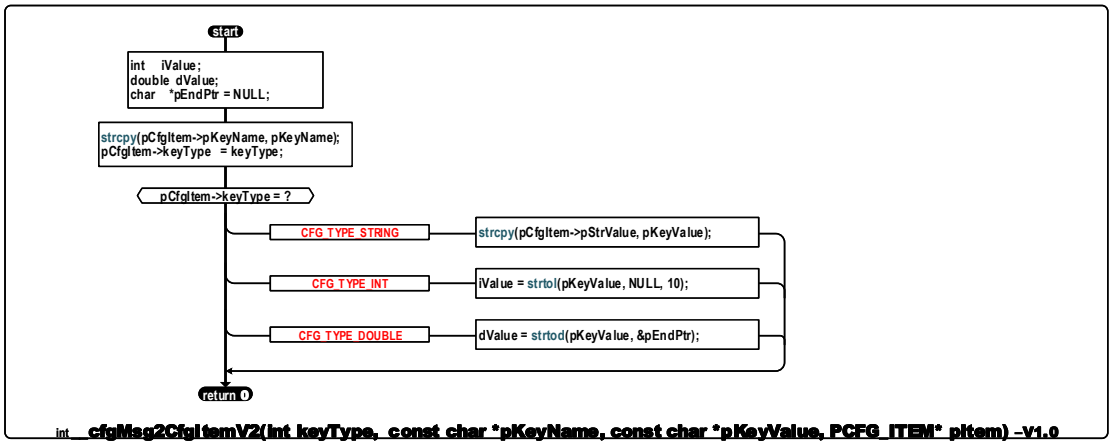

图2.12： RPC API 消息转配置项流程图

4.3.9 __cfgStartSyncGlobalCfgSvr()

启动高速缓存和数据引擎同步服务。

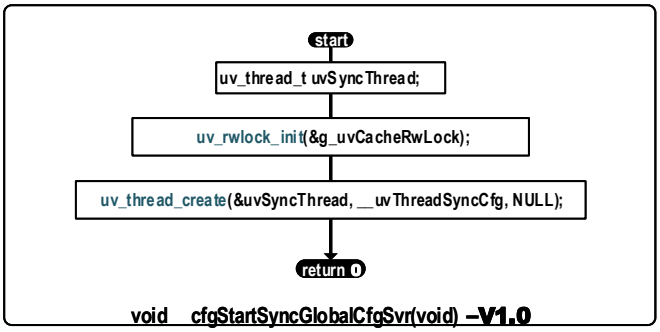

图2.13: 同步服务线程创建流程图

4.3.10 __uvThreadSyncCfg()

创建异步线程执行数据同步服务。

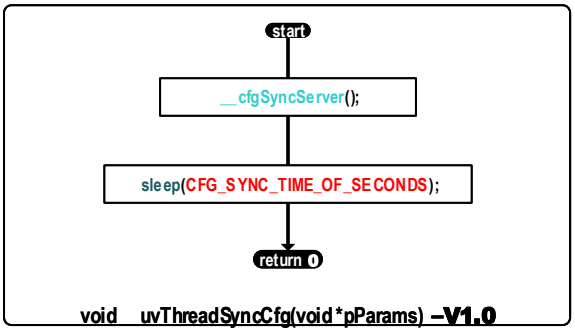

图2.14：　　同步服务线程流程图

4.3.11 __cfgSyncServer()

数据同步服务函数。

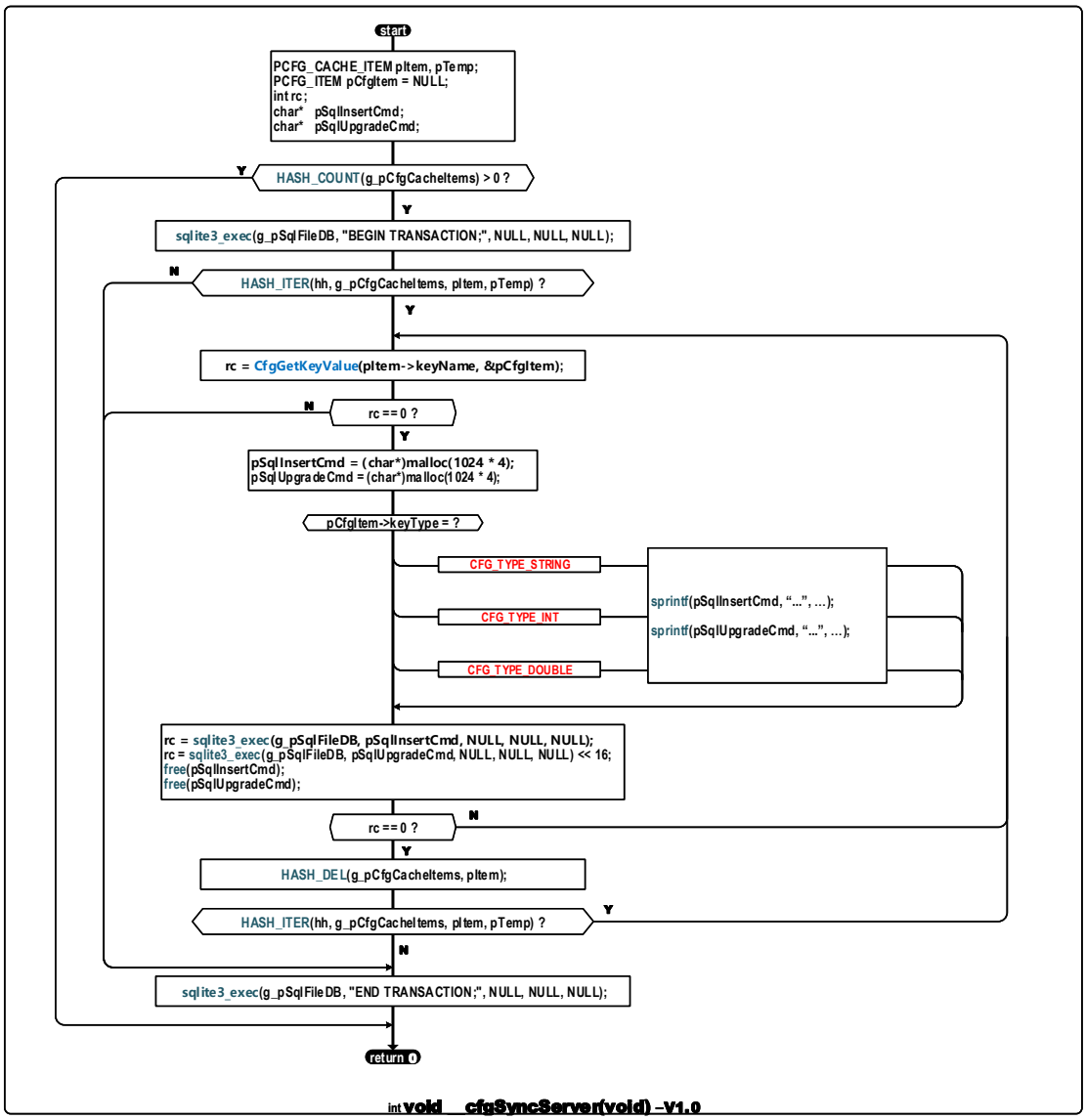

图2.15：　　高速缓存同步流程图

5 Review Information

Topic of Review: **Date**:

Related Documents:

Meeting Moderator:

Meeting Scribe:

Invitees	Attended (Y/N)

Approval required from anyone who didn't attend?

If so, from whom?

Agenda/Issues discussed:

1.
2.
3.
4.

Outcome:

Action items

Item	Owner	Date Due	Status	Closed?